

SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

SELF-ROUTE EXPANDABLE MULTI-MEMORY PACKET SWITCH WITH DISTRIBUTED SCHEDULING MEANS

Background of the Invention

[0001] FIELD OF THE INVENTION

[0002] The present invention relates to the transmission of data packets such as ATM packets between Local Area Networks (LAN) interconnected by a switch engine and relates in particular to a data transmission system including a self-route expandable multi-memory packet switch wherein the scheduling means is distributed.

[0003] BACKGROUND OF THE INVENTION

[0004] Local Area Networks (LAN) such as Ethernet or token-ring networks, are generally interconnected through hubs. The hub is a system consisting of LAN adapters that communicate together through a switch card containing a switch engine. Such a switch engine can be either a shared memory switch or a crossbar switch.

[0005] The shared memory switch is a device wherein the packets received by the input ports are stored into a memory at locations the addresses of which are determined by queues containing the packet destination addresses, the packets being transmitted on the output ports as the destination addresses are dequeued. Although such a switch enables to incur a very low cell-lost rate, it presents a bottleneck due to the requirement of the memory bandwidth, the segregation of the buffer space and the centralized control of the buffer which causes the switch performance to degrade as the size of the switch increases. A traditional approach to design a large shared

memory switch has been to first design a feasible size shared memory switch and then to interconnect a plurality of such modules in order to build a large switch. This general scheme of switch growth is known to cause degradation in performance of shared memory architecture as the switch grows in size insofar as the memory access controller will have to increase the number of all centralized control functions and memory operations thereby reducing drastically the access to the shared memory. A growable switch approach packet switch architecture is a plurality of shared memory switches organized in a single stage preceded by a buffer-less interconnection network. This approach does not allow global sharing of memory space along all its inputs and outputs. It is known that this approach does not provide the best buffer utilization as possible for a buffer belonging to a group of output ports to overflow under unbalanced or bursty traffic conditions.

[0006] The other technique, the crossbar switch, does not use a shared memory to store the data packets. In such a switch, the data are stored in the adapters and the switching data connection is established by sending requests to a centralized scheduler which determines whether it is possible to satisfy the requests. For this, the scheduler includes an algorithm unit which determines the best data connection to establish at each time. Such a determination is based upon the selection of the request amongst all requests received from the LAN adapters which meets some predetermined criteria such as a priority order, the selection of unicast/multicast, the selection between reserved bandwidth data and non-reserved bandwidth data, or any other criteria defined by the user.

Brief Summary of the Invention

[0007] The main drawback of the prior art is that the use of a centralized scheduler must know the complete switching topology of the system. If the switch grows in size by increasing the number of input and output ports, it is required to redesign the centralized scheduler. Furthermore, a speed expansion is also impossible without redesigning the centralized scheduler.

[0008] Accordingly, the main object of the invention is to provide a packet switch module wherein the scheduling function is not centralized but distributed between all output ports thereby enabling a port expansion without requiring a scheduler redesign.

[0009] The invention relates therefore to a data transmission system comprising a plurality of Local Area Networks (LANs) interconnected by a hub including the same plurality of LAN adapters respectively connected to the LANs and a packet switch comprising at least a packet switch module interconnecting all LAN adapters wherein a packet transmitted by any adapter to the packet switch includes a header containing at least the address of the adapter to which the packet is forwarded, a switch module comprising a plurality of input ports and a plurality of output ports both being respectively connected to the LAN adapters, and each couple of an input port and an output port defining a cross point within the switch module. The system comprises a memory block located at each cross point of the switch module for storing any data packet which is received from the input port corresponding to the cross point and which is to be forwarded to the output port corresponding to the cross point, and a scheduler associated with each output port for selecting at each clock time a memory block among all memory blocks corresponding to the output port and causing the memory block to forward the stored data packet to the output port when predetermined criteria are met.

Brief Description of the Several Views of the Drawings

[0010] The above and other objects, features and advantages of the invention will be better understood by reading the following more particular description of the invention in conjunction with the accompanying drawings.

[0011] FIG. 1 is a schematic block diagram of a data transmission system including four LANs interconnected by a hub according to the principles of the invention.

[0012] FIG. 2 represents schematically a data packet with the header of two bytes added by the adapter which is transmitted through a packet switch according to the invention.

[0013] FIG. 3 is a block diagram representing the features of the packet switch being used in the packet data flow.

[0014] FIG. 4 is a block diagram representing an input control block of the packet switch.

[0015] FIG. 5 is a block diagram representing a memory block located at each cross point

of the packet switch.

[0016] FIG. 6 is a block diagram representing an input expansion data block of the packet switch.

[0017] FIG. 7 is a block diagram representing an output data block of the packet switch.

[0018] FIG. 8 is a block diagram representing the complete architecture of the packet switch

[0019] FIG. 9 is a flow chart representing the steps controlled by the scheduler when a single or multiple overflow occurs.

Detailed Description of the Invention

[0020] The invention is implemented in an environment illustrated in FIG. 1 wherein a plurality of Local Area Networks (LAN) 10-1, 10-2, 10-3, 10-4 are interconnected together by hub 12 including packet switch 14. The Local Area Networks may be of the type ATM, Ethernet, or token-ring. Each LAN is connected to packet switch 14 in hub 12 by means of LAN adapter 16-1 for LAN 10-1, 16-2 for LAN 10-2, 16-3 for LAN 10-3 and 16-4 for LAN 10-4. Each adapter 16-1 to 16-4 is connected to packet switch 14 by means of data bus in 13 (bus 13-1 to 13-4) and data bus out 15 (bus 15-1 to 15-4). Connected to packet switch 14 are input expansion bus 17 and output expansion bus 18 which are respectively used for increasing the number of input ports and the number of output ports as explained hereafter.

[0021] Data bus in 13 carries the data packets coming from the input adapter and data bus out 15 carries the outgoing data packets to the output adapter. As explained hereafter, each incoming packet includes a self-routing header inserted by the adapter, this header being used to independently process the data packet through the different stages of the switch module.

[0022] FIG. 2 represents the format of the data packets exchanged between the LAN adapters through the packet switch. It is assumed that the data are packets of 53 bytes. A header of 2 bytes is added to each packet by the adapter. The first byte of the header is composed of an identification field of three bits (bits 0-2) and a module address field of 5 bits (bits 3-7). The second byte of the header is used in the unicast

configuration and gives in bit map the destination output port selection.

[0023] General data flow structure

[0024] In reference to FIG. 3, the general data flow structure of switch module 14 according to the invention, is composed of a plurality of input bus like data bus in 13 respectively connected to the input ports of the switch and a plurality of output bus like data bus out 15 respectively connected to the output ports of the switch.

[0025] For each cross point such as the cross point defined by data bus in 13 and data bus out 15, there are an input control block 100, a memory block 200, an input expansion data block 300 and an output control block 400. Input control block 100 is common for all memory blocks which correspond to data bus in 13 and output control block 400 is common for all memory blocks which correspond to data bus out 15. Input expansion data block 300 is connected in input to input expansion bus 17 and is common to all memory blocks which correspond to data bus out 15. All the memory blocks corresponding to data bus in 13 are connected to a distributed data bus 50 itself connected to output expansion bus 18 by means of a gate 36. All the memory blocks corresponding to data bus out 15 are connected to output data bus 60 and to overflow data bus 70, the function of which will be explained later.

[0026] The data packets which are received by each memory block 200 from input control block 100 are analyzed and stored into memory, and are then released to output control block 400 through output data bus 60. Then, the data packets are sent by output control block 400 over data bus out 15. All these operations are synchronized and controlled by scheduler 500 within output control block 400 by means of control lines such as lines 206, 236 and 242.

[0027] As illustrated in FIG. 4, input control block 100 comprises principally data bus in 13 for receiving data packets and means for storing the incoming data packets according to their destination and releasing these packets into distributed data bus 50. Such means include buffer 120 for buffering and validating the data packet received from input bus 104 and input memory unit 122 for storing the data packets under the control of memory control block 114. The input memory unit is preferably a memory adapted to store a plurality of data packets, the write signal being sent by

memory control block 114 after validation of the data in buffer 120. When a data packet is forwarded over distributed bus 50, a read signal is sent to memory control block 114 enabling memory control block 114 to know the filling level of input memory unit 122. Assuming that input memory unit 122 is full, the data packet within buffer 120 is not allowed to be transferred into input memory unit 122 and an overflow signal is forwarded to a scheduler on line 236 as described hereafter.

[0028] As described later, several modules can be grouped together to constitute the packet switch. For this, it is necessary to have multiplexer 116 between data bus in 13 and distributed data bus 50. Input control signal 118 coming from rank selector 800 determines the selection of the input to the multiplexer. In case of several switch modules, only the data packets received by the first module must be buffered to avoid the risk of overflow. In such a case, the multiplexer input selected by control signal 118 is the output of input memory unit 122 for the module 0 wherein data bus in 13 and following bus 106 is directly connected to distributed data bus 50 by multiplexer 116 for the subsequent modules. Note that the output of input memory unit 122 is also selected if there is only one switch module in packet switch 14.

[0029] FIG.5 shows memory block 200 composed of memory select block 244, header detection block 210, header configuration setting and validation control block 212, memory controller 234, data memory unit 226, data selector block 238, and header validation control block 216.

[0030] Header configuration setting and validation control block 212 has the functions of storing the module rank from rank selector 800, storing the configuration data memory address from configuration interface mechanism 600, analyzing the data packet type (multicast, unicast, etc.), and authorizing (or not) the reception of the incoming data packet according to the destination data packet address.

[0031] a) At initialization time, header configuration setting block 212 receives the switch module rank from rank selector 800 through bus 118. The module rank is needed for determining the global physical address of each output port of the switching system. Each header configuration-setting block attached to the same column output port has the same decoding address. Assuming that each switch module is an 8x8 port module, the 1st column corresponding to the output port 1 has the decoding address

'0'; the 2nd column has the decoding address '1' and so on until the column 8. Note that the switch module could be an m x m port module with m different from 8.

[0032] If the switch module is single, then the decoding address on each column is unchanged. But, in port expansion with several modules interconnected together, the 1st column of modules has to decode the address range (0-7), the 2nd column of modules has to decode the address range (8-15), the 3rd column of modules has to decode the range address (16-23), and so on until the last column of modules. If there are n columns of modules, the block 212 assigns an offset of 8 x k to the output port address in the module, with k being 0 to n-1.

[0033] b) The second function of the header configuration and setting block 212 allows modifications of the pre-set internal output port memory address through the configuration interface. This function is used in internal speed expansion mode, where 2 or more output ports or columns have to be combined in order to grow the data throughput of the port. Configuration interface mechanism 600 configures the memory block through configuration bus 204.

[0034] c) The third function of the header configuration and setting block 212 is to detect whether the packet is a multicast address packet. If so, the header of the packet has a specific configuration determining that all the following packets, which have all a specific header, are the packets of a multicast frame. In such a case, header configuration and setting block 212 analyzes also the 54 bytes of the packet following the header to determine whether the output port associated with the memory block corresponds to one of the output ports to which the multicast frame is addressed.

[0035] d) Header detection block 210 defines the start of each incoming data packet. This block receives clocking signal through the signal 208 at each clock time.

[0036] e) Header validation control block 216 uses control signals from block 212, block 210, and validation signal 206 from scheduler 500, to authorize memory controller 234 to store the incoming data packet into data memory unit 226.

[0037] f) Data validation block 244 selects either distributed data bus 50 or overflow data bus 70 depending on control signal 248 driven by scheduler 500. By default, distributed data bus 50 is connected to data memory unit 226 until an overflow is

detected.

[0038] g) Data memory unit 226 stores and releases the data packets under the control of memory controller 234.

[0039] h) Data Memory controller 234 performs the functions of controlling the address release, enqueue and dequeue mechanisms, generating read and write signals, and generating memory overflow signal 236 to scheduler 500.

[0040] i) Overflow data bus 70 (one per output), is connected to all memory blocks, along internal output data bus 60 in order to reassign the overflow data packet to another memory block. For this, scheduler 500 activates signal 242 controlling overflow connection block 238 which can be an AND circuit connecting distributed data bus 50 to overflow data bus 70 through bus 240. Scheduler takes the decision after receiving flow controls signals 236 from memories connected on the same output port. The decision is to determine the usable memory wherein the overflow data packet can be stored. This is particular useful, due to the fact that the data packet is re-routed to another memory block of the same output port.

[0041] FIG. 6 shows input expansion data block 300 which is composed of header processing block 302, header validation block 308, expansion memory unit 312, and memory controller 314.

[0042] Input expansion bus in 17 connected to header processing block 302 carries the data packet coming from another switching module in expansion mode. Header processing block 302 is also connected in input to overflow data bus 70 for receiving an overflow data packet. Header processing module 302 is connected in output to header validation block 308 by data bus 306. The function of the header processing block is to select the appropriate data bus, according to the configuration mode line 320 from rank selector 800. This line carries the necessary module rank information.

[0043] The header validation block 308 receives control signal validation 206 coming from the scheduler 500. Header validation block 308 signals an incoming data packet to memory controller 314 through control signal 324 and sends the data packet to memory block 312 through data bus 310.

[0044] The main function of expansion memory unit 312 is to store the incoming data packet coming from the expansion data bus or from the overflow data bus, under the control of memory controller 314 which controls the write/read operations to the memory, and generates memory flow control signal 236 to scheduler 500.

[0045] FIG. 7 shows output data block 400 which is composed of data selection block 402, output memory unit 406, and memory controller 408.

[0046] The function of output data block 400 is to receive data packets from internal output bus 60, to validate data packets from internal output bus 60, to store into output memory unit 406 the incoming data, and to release data packet on data bus out 15.

[0047] The function of data selection block 402 is to receive internal output data bus 60, to validate the incoming data packet when receiving validation signal 206 coming from the scheduler, and to activate validation data signal 410 to memory controller 408.

[0048] Output memory unit 406 connected to data selection block 402 by data bus 404, stores incoming data packets under the control of memory controller 408. The function of the latter is to store the incoming data packets into the memory block, to release data packets from the output memory unit, to control the storing memory address, and to generate flow control signal 236 to scheduler 500.

[0049] The data packets after being released from output memory unit 406 by the memory controller, are sent over output data bus 15.

[0050] Scheduler (500)

[0051] An essential feature of the invention is to use a scheduler, as illustrated in FIG. 3 and FIG. 8, for each output port of the packet switch rather than a centralized scheduling mechanism as in a crossbar switch.

[0052] The main functions of scheduler 500 are to receive the necessary information coming from all attached column memory blocks 200, to activate the validation of the incoming data packet for the selected memory block, to determine the outgoing data packet by choosing the memory block according to a round-robin mechanism which

can be based on priority selection and/or any other selections, to control the memory overflow, to flow control the output ports, and, to report flow control signals 710 to overflow control mechanism 700, and therefore alert back pressure mechanism 900.

[0053] Rank selector (800)

[0054] Rank selector 800 located in the bottom right corner of FIG.8 is a mechanism using a number of input pins hardwired on the board, that define the module rank in a packet switch including a plurality of switch modules.

[0055] In the case of single module, this address is '0'. In the case of port expansion, many switch modules may interconnect together. For the ease of comprehension, it is assumed a 16x16 switch system configuration using four 8x8 switch modules. The 2 modules of the 1st column of modules have to be hardwired to '0'. The 2 other modules of the 2nd column of modules have to be hardwired to '1'. The same algorithm applies for an N x N switch system configuration.

[0056] The physical destination address known by the adapters is the final destination address and is contained in the header of each of the data packets.

[0057] Overflow control

[0058] Based upon the overflow signals coming from all memory blocks on lines 236 as illustrated in FIG. 3 and FIG.8, the scheduler determines the memory blocks which overflow during each data packet time (internal clock time for storing one data packet).

[0059] As illustrated by the flow chart of FIG. 9, the scheduler first checks whether there is a memory block which overflows (step 30). If so, it is then checked whether it is a multiple overflow (step 32). In case of multiple overflows, the scheduler uses a selection algorithm (generally a round robin selection) to select the appropriate memory block which can receive the next data packet (step 34). Of course, if it is a single overflow, the step of memory selection is not necessary. In such a case or after the selection, the memory block which overflows is disabled by the scheduler on line 206 (step 36) and a usable memory block is enabled by the scheduler on line 248 (step 38). Then, overflow bus 70 is enabled by line 242 from the scheduler to carry

the data packet into the data memory unit of the memory block which is validated by line 248 (step 40). When there is no memory overflow (step 30) or after the transfer of the data packet over overflow bus 70, the process is ended (step 42).

[0060] It must be noted that such an overflow processing by a scheduler associated with each output port, presents the advantages of flow controlling the internal data, avoiding the loss of the data packet, having a better distribution of the data packets, and delaying the generation of a back pressure signal as described hereafter only when it is not possible to process the memory overflow normally.

[0061] Configuration interface mechanism (600)

[0062] The configuration interface mechanism 600 located on bottom left of FIG. 8 is the mechanism controlling the configuration of each column output port.

[0063] Assuming that the switch is an 8x8 output port, at the end of the Initialization, the 1st column corresponding to the output port 1 has the decoding address '0'. The 2nd column has the decoding address '1' and so on until the column 8. The configuration interface mechanism allows the traffic management to modify the address of each column. As an example the packet switch may have the following configuration:

[0064] Port_1: Decoding address '0'

[0065] Port_2: Decoding address '1'

[0066] Port_3: Decoding address '2'

[0067] Port_4: Decoding address '2'

[0068] Port_5: Decoding address '4'

[0069] Port_6: Decoding address '5'

[0070] Port_7: Decoding address '6'

[0071] Port_8: Decoding address '6'

[0072] This function is used to increase the Internal Speed. The Port_3 and Port_4 decode

the same incoming data packet, which improves the performances of the adapter. The same applies as Port_7 and Port_8.

[0073] Configuration interface mechanism 600 sends information through bus 204 to the configuration setting and detection block 212 of each memory block of each output port (see FIG. 5). Configuration interface mechanism 600 receives information through bus 610 from traffic management. In the case of port expansion (several modules interconnected together), each module is connected to bus 610.

[0074] The traffic management delivers through bus 610 the information about the module physical address, the row/column physical address, and the modified address of the row/column data memory block. The traffic management accesses only one configuration interface 600 at a time.

[0075] Back-pressure Mechanism (900)

[0076] The back-pressure mechanism 900 located in the top left corner of the FIG.8, has the functions of receiving flow control bus 910 from overflow control block 700, generating flow control bus 915 to overflow control block 700, receiving flow control information on bus 924 from the right adjacent switch module, receiving flow control information on bus 925 from the bottom adjacent switch module, generating flow control information on bus 922 to the left adjacent switch module, and generating flow control information on bus 923 to the top adjacent switch module.

[0077] Of course, in a single module configuration there is no information exchanged with other modules. Bus 922, from back-pressure mechanism 900 connected to the input ports, is made of n independent signals, with one signal per input port.

[0078] The generation of a back-pressure signal to the adapters is to stop (or reduce) the flow of the data packets transmitted to the packet switch when there is too much overflow detected by one or several schedulers. The back-pressure signals are generated after receiving flow control information from overflow mechanism 700 through bus 910.

[0079] When a memory block is not able to store any more of the data packets, an overflow control signal is sent to the corresponding scheduler through bus 236. Each

scheduler alerts overflow mechanism 700 through control bus 710. Overflow mechanism 700 receives overflow control signals from all schedulers and informs back-pressure mechanism 900 through bus 910 to back-pressure the corresponding adapters.

[0080] In port expansion configuration, back-pressure mechanism 900 receives overflow information from the right adjacent switch module, and from the bottom adjacent switch module, and back-pressure mechanism 900 generates overflow information to the top adjacent switch module.

[0081] When back-pressure mechanism 900 receives overflow information from the bottom adjacent switch module, back-pressure mechanism 900 informs overflow mechanism 700 through bus 915, which in turn alerts corresponding schedulers 500 through bus 710 and requests schedulers 500 to decrease the transmission of the data packets.

[0082] When back-pressure mechanism 900 receives overflow information from the right adjacent switch module, back-pressure mechanism 900 alerts the corresponding input adapters through bus 922 and requests the input adapters to decrease the transmission of the data packets.

[0083] Although specific embodiments of the present invention have been illustrated in the accompanying drawings and described in the foregoing detailed description, it will be understood that the invention is not limited to the particular embodiments described herein, but is capable of numerous rearrangements, modifications and substitutions without departing from the scope of the invention. The following claims are intended to encompass all such modifications.